关卡 2：牛刀小试

课后作业

华为技术有限公司

理论

课堂思考题（20 分）

讨论 1：请描述 MindSpore 的基础数据处理流程。

答案：（提交 pr 作业）

- Shuffle： 对数据集进行混洗，随机打乱数据顺序。
- Map：将指定的函数或算子作用于数据集的指定列数据，实现数据映射操作。
- Batch：将数据集分批，分别输入到训练系统中进行训练，可以减少训练轮次，达到加速训练过程的目的。
- Repeat：对数据集进行重复，达到扩充数据量的目的。
- Zip：将两个数据集进行列拼接，合并为一个数据集。

讨论 2：定义网络时需要继承哪一个基类?

答案：mindspore.nn.Cell

讨论 3：定义网络时有哪些必须编写哪两个函数?

答案：__init__()，construct()

讨论 4：讨论 3 中提到的两个函数有什么用途?

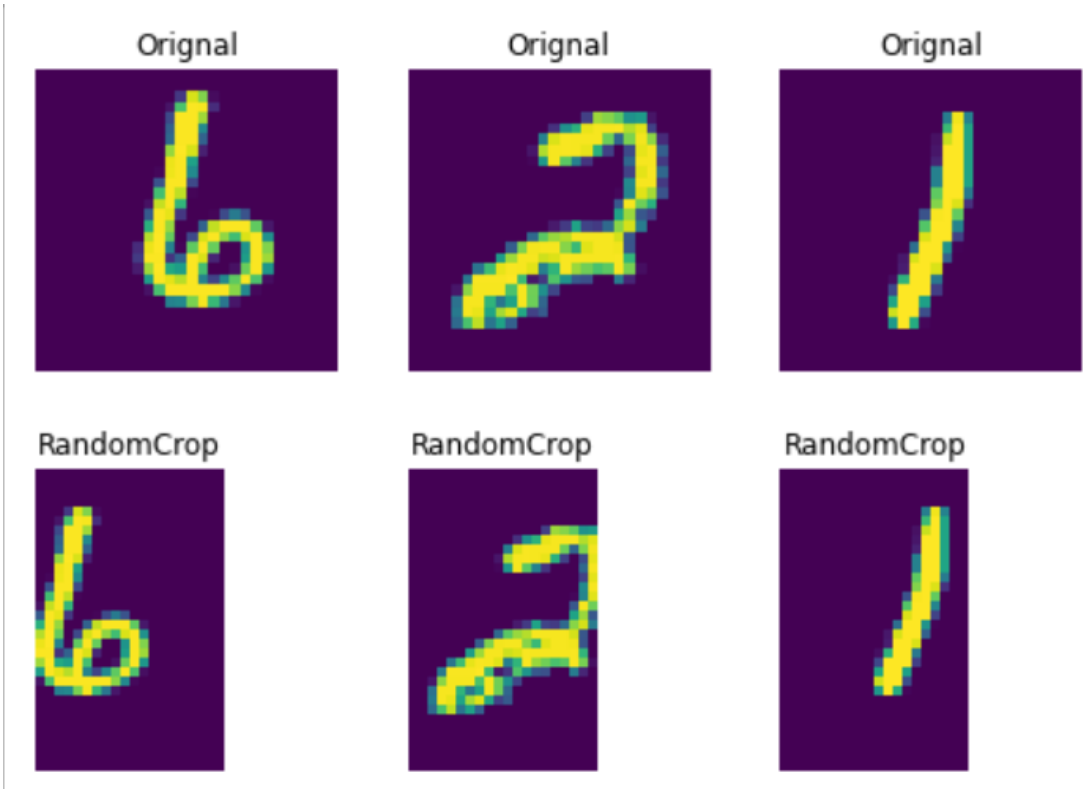
答案：一般会在__init__()中定义算子，然后在 construct()中定义网络结构。
__init__()中的语句由 Python 解析执行；construct()中的语句由 MindSpore 接管，有语法限制；

实验

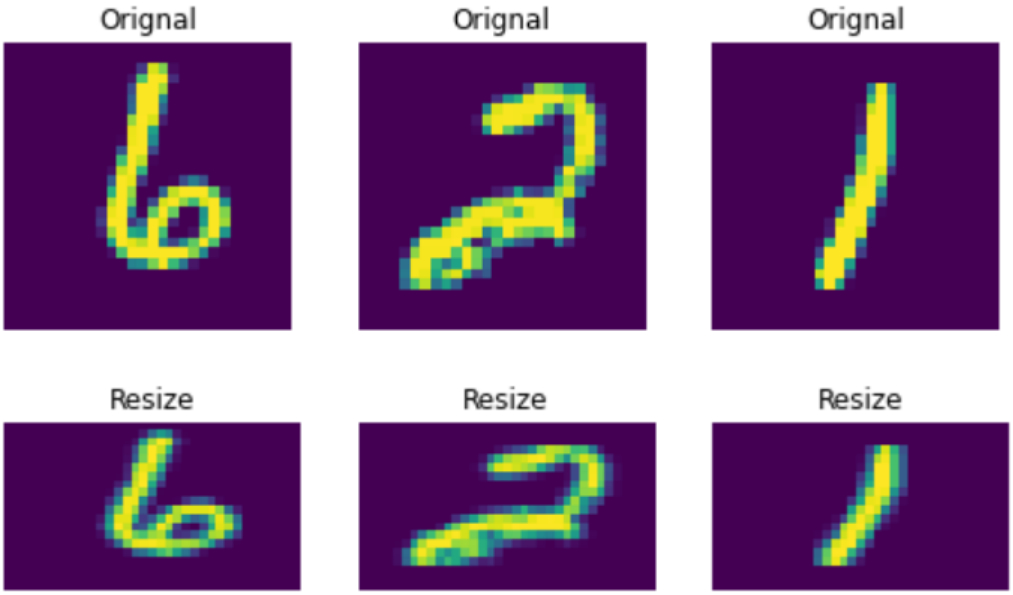
1. MindSpore 进阶操作实验（30 分）

任务 1：请使用自选彩色图片进行图像预处理和增强操作，至少包含随机裁剪、调整图片大小、像素归一化、随机翻转、通道转换操作，并返回操作前后对比结果。

1. 随机裁剪：



2. 调整图片大小：



3. 像素值修改：

步骤 3 像素值修改

mindspore.dataset.vision.c_transforms.Rescale(rescale, shift)

In [13]:
```
import mindspore.dataset.vision.c_transforms as CV

# 从mnist dataset读取3张图片
mnist_dataset = ds.MnistDataset(dataset_dir=dataset_dir, num_samples=3)

# 原图像素值
for dic in mnist_dataset.create_dict_iterator(output_numpy=True):
    print("原图像素值:", dic['image'][:,:,0])

# rescale 缩放系数，shift 平移大小，相当于ax+b
rescale = CV.Rescale(rescale=3, shift=100)
mnist_dataset1 = mnist_dataset.map(operations=rescale, input_columns=["image"])

# 修改后的像素值
for dic in mnist_dataset1.create_dict_iterator(output_numpy=True):
    print("修改后的像素值:", dic['image'][:,:,0])
```

```
 209 193 130 139  14   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  36 129 164 247 252
  252 253 226 244  49   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0 112 241 253 252 252 252
  199  83 205 251 212  30   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  15 225 252 253 252 247 162
   49   0  57 246 252  77   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   4 138 252 191 112  38   0
    0   0  85 252 252  42   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0  50 245 253  86   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0 121 252 252  42   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   4 116 244 221  65   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0  69 252 199 162   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0 148 217  69  49   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  18  43 148  51  43
```

4. 图像标准化：

步骤 4 图像标准化

mindspore.dataset.vision.py_transforms.Normalize(mean, std)

将输入的NumPy图像数组（形状必须为(C, H, W)）用给定的均值和标准差进行标准化。 数组的值必须在(0, 1]范围内。

In [14]:
```
import numpy as np
import mindspore.dataset as ds

# 读取图片，decode=TRUE 解码为（H,W,C）的格式
dataset = ds.ImageFolderDataset('./data/cat', decode=True)

# 原图像素值
for x in dataset.create_dict_iterator():
    print("原图像形状:",x['image'].shape) # 高，宽，通道数
    print("原图像素值:",x['image'])

# 进行标准化操作
normalize_op = CV.Normalize(mean=(123.68,116.78,103.94), std=[1, 1, 1])
dataset = dataset.map(operations=normalize_op, input_columns=["image"])

# 标准化之后的像素值
for x in dataset.create_dict_iterator():
    print("标准化之后的像素值:",x['image'])
```

```
原图像形状: (2448, 3264, 3)
原图像素值: [[[131 145 109]
```

5. 随机水平翻转图像：

6. 转换图像通道：

步骤 6 转换图像通道

mindspore.dataset.vision.c_transforms.HWC2CHW

In [16]:
```
import mindspore.dataset.vision.c_transforms as CV
import numpy as np
import mindspore.dataset as ds

# 读取图片，decode=TRUE 解码为（H,W,C）的格式
dataset = ds.ImageFolderDataset('./data/cat', decode= True)

# 原图shape
for x in dataset.create_dict_iterator():
    print("原图形状：", x['image'].shape)

# 转置输入图像从形状(H, W, C)到形状(C, H, W)
HWC2CHW = CV.HWC2CHW()
dataset1 = dataset.map(operations=HWC2CHW, input_columns=["image"])

# 通道转换之后的shape
for x in dataset1.create_dict_iterator():
    print("通道转换之后的形状：",x['image'].shape)
```

```
原图形状： (2448, 3264, 3)
通道转换之后的形状： (3, 2448, 3264)
```

2. MNIST 手写体识别实验（30 分）

任务 1：将模型预测结果截图提交。

模型评估

查看模型在测试集的准确率

In [24]:
```
model.eval(test_data) # 测试网络
```

Out[24]:
```
{'accuracy': 0.9872}
```

效果展示

In [25]:
```
data_path=os.path.join('data', 'test')

ds_test_demo = create_dataset(test_path, batch_size=1)

for i, dic in enumerate(ds_test_demo.create_dict_iterator()):
    input_img = dic['image']
    output = model.predict(input_img)
    predict = np.argmax(output.asnumpy(),axis=1)[0]
    if i>9:
        break
    print('True: %s, Predicted: %s'%(dic['label'], predict))
```

```
True: [0], Predicted: 0
True: [2], Predicted: 7
True: [4], Predicted: 4
True: [7], Predicted: 7
True: [5], Predicted: 5
True: [1], Predicted: 1
True: [0], Predicted: 0
True: [5], Predicted: 5
True: [5], Predicted: 5
True: [6], Predicted: 6
```